Dataflow program implementation onto a heterogeneous multiprocessor platform

Kevin Martin, Jean-Philippe Diguet, Emmanuel Casseau, Yaset Oliva

Dataflow program implementation onto a heterogeneous multiprocessor platform

Kevin Martin, Jean-Philippe Diguet
Lab-STICC, UMR 3192, University of Bretagne-Sud, France
{kevin.martin,diguet}@univ-ubs.fr

Emmanuel Casseau, Yaset Oliva
IRISA/INRIA, University of Rennes 1, France
{emmanuel.casseau,yaset.oliva-venegas}@irisa.fr

***Abstract*—This paper presents an implementation of dataflow programs specified in RVC-CAL onto a heterogeneous multiprocessor platform based on the Xilinx Zynq platform. At first, the network of actors is mapped onto the ARM processor and is executed. A new mapping is then performed onto the MicroBlaze processors thanks to metrics computed at runtime.**

I. Dataflow program

Dataflow programming is a convenient way to specify an application. It explicitly expresses both spatial and temporal parallelism. In nowadays context of multicore and multiprocessor era, where making use of the available resources is a major concern, the dataflow approach fits naturally. This work relies on the RVC-CAL model [1], a subset of CAL used for video coding and standardized by the MPEG working group. In this model, an application is represented as a directed graph where the nodes represent the *actors*, the computational units, and the edges represent the communication channels. Figure 1 presents a network of actors. Each actor consumes and produces data (called token) from and to the channels and is characterized by a *Model of Computation* (MoC), which defines the behavior of the actor. Mainly, an actor can be static or dynamic. In the original model, the channels are based on FIFO and may be unbounded. The implementation basically forces to bound these channels, and some of the MoC used in RVC-CAL allow an out-of-order access to the token.

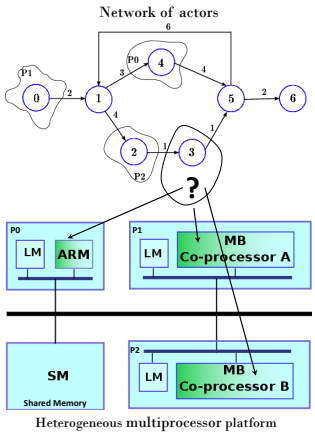

Fig. 1. Mapping a network of actors onto a heterogeneous platform

II. Heterogeneous multiprocessor platform

In this work, the multiprocessor platform relies on the Xilinx Zynq platform. The Zynq is composed of a fixed processing system containing an ARM dual-core Cortex-A9 MPCore processor and several peripherals, and a programmable logic area that can be used to implement custom logic and soft-cores. MicroBlaze processors are used in this work. Figure 1 presents a platform composed of an ARM processor and two MicroBlaze processors (MB). Each MicroBlaze may be customized adding a co-processor that can accelerate the execution of an actor. To avoid memory relocation, a shared memory is used to implement the FIFOs, i.e. all the FIFOs are in the same memory. The components are connected through an AXI bus.

III. Actor mapping

Starting from the dataflow network, it is required to map the actors onto the available processing units of the execution platform. This process defines the set of actors a processor executes. The mapping process can be translated to a graph partitioning problem. It is a NP-complete problem so heuristics have to be used in practice to compute the mapping at runtime. Executing efficiently programs on a multiprocessor platform requires to balance the computational load over the processors. However, the mapping is not so easy because the connectivity between actors as well as the communication load and characteristics of the platform such as processor types and communication paths also need to be taken into account. In our work, the network of actors is first mapped onto one single processor, the ARM core, to calculate functional metrics. We consider the computational load, which represents the individual total work of each actor in terms of execution time, and communication load, which describes how much an actor communicates with others in terms of the number of tokens transferred in a time interval. The mapping technique proposed in [2] is used to re-partition the graph in such a way every partition of a graph (also called a cluster) is mapped onto one MicroBlaze. A cluster-level scheduler is used to schedule the actors of a given cluster. This scheduler sequentially tests the firing rules of the actors, and fires an actor if a firing rule is valid. The simple round-robin scheduling strategy is used [3], based on a compile-time ordering of the actor execution. This strategy is easy to implement and usually provides high data-rates thanks to spatial/temporal locality.

References

[1] M. Mattavelli, J. W. Janneck, and M. Raulet, "Mpeg reconfigurable video coding," in *Handbook of Signal Processing Systems*. Springer US, 2010.

[2] T. D. Ngo, K. Martin, and J.-P. Diguet, "Communication-model based embedded mapping of dataflow actors on heterogeneous MPSoC," in *DASIP*, 2014.

[3] H. Yviquel, E. Casseau, M. Wipliez, and M. Raulet, "Efficient multicore scheduling of dataflow process networks," in *Signal Processing Systems (SiPS), 2011 IEEE Workshop on*, 2011, pp. 198–203.